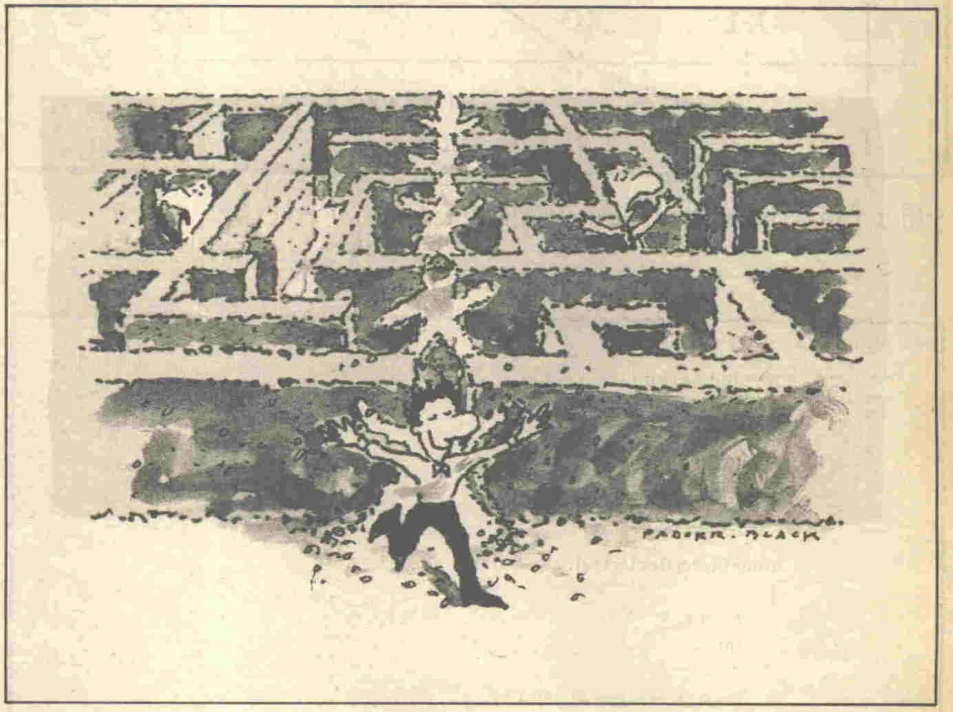
ซอฟต์แวร์
ขจร โรจนเมธินทร์

แนะนำวิธีการหารากของสมการเชิงโค้ง

เป็นการนำเอาคอมพิวเตอร์เข้ามาช่วยจัดการกับปัญหาทางคณิตศาสตร์ในการแก้สมการหรือหารากของสมการ



คุณคงเป็นคนหนึ่งที่เคยเรียนวิชาคณิตศาสตร์ คิดว่าอย่างน้อยก็ต้องเคยเรียนสมัยที่อยู่ชั้นประถมแน่นอน หลายท่านก็อาจจะชอบวิชานี้ แต่หลายท่านเห็นวิชาคณิตศาสตร์เป็นเสมือนยาขมหม้อใหญ่เลยทีเดียว วันนี้ผมจะขอหยิบเอาบางส่วนของคณิตศาสตร์มาพูดคุย โดยใช้คอมพิวเตอร์มาช่วยจัดการกับปัญหาทางคณิตศาสตร์ การแก้สมการหรือการหารากของสมการเป็นสิ่งที่ต้องพบเสมอในวิชาคณิตศาสตร์ บางสมการก็สามารถหารากออกได้ง่าย แต่บางสมการก็ทำเอาปวดหัวได้เหมือนกัน วิธีการหารากของสมการนั้นมีหลายวิธี แต่ในบทความนี้ขออนุญาตเสนอวิธีที่ทำได้ง่ายสัก 3-4 วิธี

วิธี Bisection

คุณคงจำได้ว่าถ้าพูดถึงฟังก์ชันทางคณิตศาสตร์นั้นมักจะใช้ตัวแปร x มาอธิบายถึงฟังก์ชันเหล่านั้น เช่น $f(x) = x^2+4x+4$ เป็นต้น สำหรับวิธี Bisection นี้จะใช้หารากของสมการเชิงโค้งที่ $f(x) = 0$ โดย $f(x)$ นั้นต้องเป็นฟังก์ชันต่อเนื่องในช่วงหนึ่ง

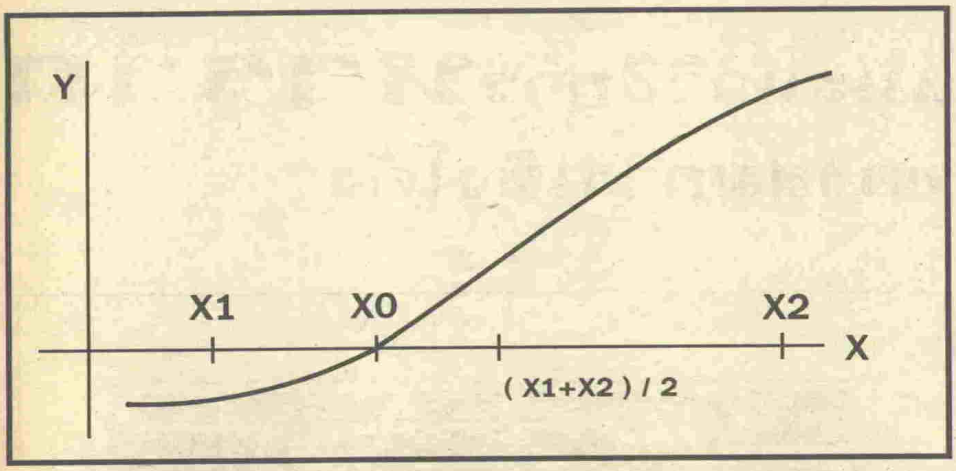
จากรูปที่ 1 ในช่วง x1 ถึง x2 นั้น $f(x)$ เป็นฟังก์ชันต่อเนื่อง และรากของสมการคือ x0 อยู่ระหว่าง x1 ถึง x2 นอกจากนี้ค่า $f(x1)$ และ $f(x2)$ ต้องมีเครื่องหมายต่างกันดังนั้นจากรูปที่ 1 ถ้าเราหาค่าฟังก์ชันที่ $(x1+x2)/2$ ค่าที่ได้ก็อาจเป็นได้ 3 กรณีคือ

1. $f(x1+x2)/2) < 0$ คือจุดแบ่งครึ่งระหว่าง x1 และ x2 นั้นไม่ใช่รากของสมการ แต่ค่ายังอยู่ใกล้ x1 เกินไป หรือรากของสมการที่เราต้องการนั้นอยู่ระหว่าง $(x1+x2)/2$ กับ x2

2. $f(x1+x2)/2) = 0$ คือ $(x1+x2)/2$ เป็นรากของสมการที่ต้องการหา

3. $f((x1+x2)/2) > 0$ คือ $(x1+x2)/2$ ยังอยู่ใกล้ x2 รากของสมการที่ต้องการจะอยู่ระหว่าง x1 กับ $(x1+x2)/2$

ดังนั้นหากเรานำค่า $(x1+x2)/2$ มากำหนดเป็นจุด x1 หรือ x2 แล้วหาค่า $(x1+x2)/2$ ใหม่ไปจนกว่าจะได้ตามกรณีที่ 2 เราก็จะสามารถหาค่ารากของสมการได้ แต่ในทางปฏิบัติถ้าทำตามวิธีดังกล่าวนี้ กว่าจะได้รากของสมการออกก็คงต้องหาค่า $(x1+x2)/2$ หลายครั้งมากเกินความจำเป็น เนื่องจาก



รูปที่ 1 แสดงกราฟของฟังก์ชัน f(x) ซึ่งใช้วิธี Bisection

```
#include <stdio.h>
#include <math.h>

float f(float x)
{
    return(3*x - exp(-2*x));
}
float bisec(float x1, float x2, float e)
{
    float x, y;
    int  i;

    puts("Bisection Method");
    puts("-----------------------------------------");
    puts(" I   X1      X2      X0    F(X)");
    puts("-----------------------------------------");
    /************ Bisection Method *************/
    i = 0;
    do {
      x = (x1+x2)/2;
      y = f(x);
      if (y < 0)
        x1 = x;
      else
        x2 = x;
      ++i;
      printf("%3d %8.5f %8.5f %8.5f %8.5f\n",i,x1,x2,x,y);
    } while (fabs(x1-x2) >= e);
    /*****************************************/
    puts("-----------------------------------------");
    return(x);
}
main()
{
    printf("\nRoot of Equation = %9.6f\n",bisec(0,1,0.0001));
}
```

รูปที่ 2 แสดงโปรแกรมคำนวณโดยวิธี Bisection

แต่ละครั้งนั้นค่า x1 และ x2 จะถูกบีบเข้ามาใกล้กันมากขึ้นเรื่อย ๆ (ใกล้กับรากของสมการด้วย) ดังนั้นเราจึงอาจประมาณว่า ถ้า x1 และ x2 มีค่าใกล้กันมาก เช่น $(x1-x2)$ มีค่าน้อยกว่า 0.0001 เราจะถือว่าค่า $(x1+x2)/2$ นั้นเป็นรากของสมการได้คำตอบที่ใกล้เคียงความเป็นจริงและใช้เวลาหาคำตอบไม่มากนัก

จากรูปที่ 2 เป็นตัวอย่างโปรแกรมที่หารากของสมการโดยวิธี Bisection ของสมการ $f(x) = 3x - e^{-2x}$ ในโปรแกรมจะใช้ลูป do { } while (fabs(x1 − x2) >= e); ตรวจสอบว่า x1 มีค่าใกล้กับ x2 หรือยัง โดยยอมให้ต่างกันได้ไม่เกินค่า e ซึ่งเท่ากับ 0.0001 ถ้าน้อยกว่านี้จะถือว่า x1 และ x2 มีค่าเท่ากัน ก็จะหยุดลูป

วิธี False Position

สำหรับวิธีนี้ต่างจากวิธี Bisection แต่คุณสมบัติของฟังก์ชันที่จะใช้วิธีนี้หารากของสมการยังคล้ายกันคือ f(x) ต้องเป็นฟังก์ชันต่อเนื่องระหว่าง x1 กับ x2 และ f(x1) กับ f(x2) ต้องมีเครื่องหมายต่างกัน

จากรูปที่ 4 จุด (x0, f(x0)), (x1, f(x1)), (x2, f(x2)) อยู่บนกราฟของ f(x) โดย $f(x0) < f(x1) < f(x2)$ ดังนั้นถ้านำจุดทั้งสามมาหาความชันจะได้ดังนี้

$$(f(x2) - f(x1)) / (x2 - x1) = (f(x0) - f(x1))/(x0 - x1) \quad ---1$$

แต่ที่ x0 นั้นค่า f(x0) = 0 ดังนั้นสมการที่ 1 จะเป็น

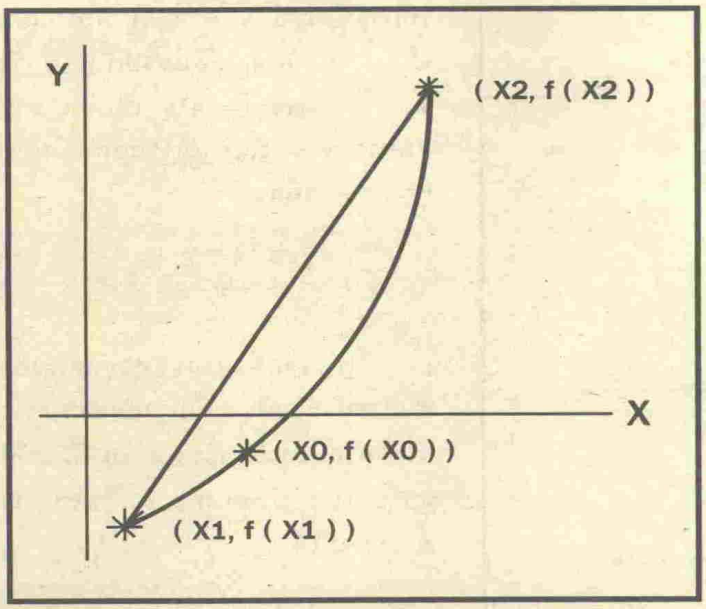
$$(f(x2) - f(x1))/(x2 - x1) = (-f(x1))/(x0 - x1)$$
$$x0 - x1 = (x2 - x1)(-f(x1))/(f(x2) - f(x1))$$
$$x0 = (x1f(x2) - x2f(x1))/(f(x2) - f(x1))$$

Bisection Method

I	X1	X2	X0	F(X)
1	0.00000	0.50000	0.50000	1.13212
2	0.00000	0.25000	0.25000	0.14347
3	0.12500	0.25000	0.12500	−0.40380
4	0.18750	0.25000	0.18750	−0.12479
5	0.18750	0.21875	0.21875	0.01060
6	0.20312	0.21875	0.20312	−0.05677
7	0.21094	0.21875	0.21094	−0.02300
8	0.21484	0.21875	0.21484	−0.00618
9	0.21484	0.21680	0.21680	0.00222
10	0.21582	0.21680	0.21582	−0.00198
11	0.21582	0.21631	0.21631	0.00012
12	0.21606	0.21631	0.21606	−0.00093
13	0.21619	0.21631	0.21619	−0.00041
14	0.21625	0.21631	0.21625	−0.00015

Root of Equation = 0.216248

รูปที่ 3 แสดงผลการคำนวณรากสมการ $f(x)=3x-e^{-2x}$ โดยวิธี Bisection



รูปที่ 4 แสดงกราฟของฟังก์ชัน f(x) ซึ่งใช้วิธี False Position

```
#include <stdio.h>
#include <math.h>

float f(float x)
{
    return(3*x - exp(-2*x));
}
float false(float x1, float x2, float e)
{
    float x, y, fa, fb;
    char  work, i;

    puts("False Position Method");
    puts("----------------------------------------");
    puts(" I    X1      X2      X0    F(X)");
    puts("----------------------------------------");
    /************ False Position Method *************/
    i = 0;
    work = 1;
    do {
     fa = f(x1); fb = f(x2);
     x = (x1*fb - x2*fa)/(fb - fa);
     if (fabs(x-x1) >= e && fabs(x-x2) >= e) {
       y = f(x);
       if (y < 0)
         x1 = x;
       else
         x2 = x;
      }
      else work = 0;
      ++i;
      printf("%2d %8.5f %8.5f %8.5f %8.5f\n",i,x1,x2,x,y);
    } while (work);
    /****************************************/
    puts("----------------------------------------");
    return(x);
}
main()
{
    printf("\nRoot of Equation = %9.6f\n",false(0,1,0.00001));
}
```

รูปที่ 5 แสดงโปรแกรมซึ่งคำนวณรากของสมการโดยวิธี False Position

จากค่า x0 ที่เราหาได้นั้นจะเห็นว่าค่า f(x0) มีโอกาสเป็นได้ 3 กรณี คือ

1. $f(x0)<0$ คือรากของสมการจะอยู่ระหว่าง x0 และ x2
2. $f(x0) = 0$ นั่นคือ x0 เป็นรากของสมการ
3. $f(x0)<0$ คือรากของสมการ

```
False Position Method
-----------------------------------------
I       X1          X2          X0          F(X)
-----------------------------------------
1       0.00000     0.25875     0.25875     0.18026
2       0.00000     0.21924     0.21924     0.01268
3       0.00000     0.21649     0.21649     0.00089
4       0.00000     0.21630     0.21630     0.00006
5       0.00000     0.21628     0.21628     0.00000
6       0.00000     0.21628     0.21628     0.00000
-----------------------------------------
Root of Equation = 0.216281
```

รูปที่ 6 แสดงผลการคำนวณรากสมการ $f(x)=3x-e^{-2x}$ โดยวิธี False position

รูปที่ 7 แสดงโปรแกรมซึ่งคำนวณรากของสมการโดยวิธี Simple iteration

```
#include <stdio.h>
#include <math.h>

float f(float x)
{
    return(exp(-2*x)/3);
}
float simple(float x, float e)
{
    float y;
    char  work, i;

    puts("Simple Iteration Method");
    puts("-----------------------");
    puts(" I        X(I)");
    puts("-----------------------");
    /************ Simple Iteration Method *************/
    work = 1;
    i = 1;
    do {
     y = f(x);
     if (fabs(y-x) >= e)
       x = y;
     else
       work = 0;
     printf("%3d       %8.5f\n",i,y);
     ++i;
    } while (work);
    /*****************************************/
    puts("-----------------------");
    return(y);
}
main()
{
    printf("\nRoot of Equation = %9.6f\n",simple(1,0.00001));
}
```

จะอยู่ระหว่าง x1 และ x0

สำหรับวิธี False Position จะเห็นว่าเราพยายามหาค่า x0 ซึ่งเป็นรากของสมการ โดยการสร้างเส้นตรงที่มีความชันให้ใกล้กับความชันของกราฟ f(x) ในช่วงที่ตัดแกน x โดยเราจะนำค่า x0 ที่หาได้มาช่วยปรับความชัน ดังนั้นเมื่อเราปรับความชันได้ใกล้กับความชันของ f(x) ค่า x0 ที่ได้ก็คือรากของสมการ

จากรูปที่ 5 เป็นโปรแกรมซึ่งใช้วิธี False position method หารากของสมการ ค่า x0 ที่หาได้แต่ละครั้งจะนำมาเปรียบเทียบกับ x1 และ x2 ว่ามีค่าใกล้เคียงกันไม่เกินค่า e หรือยัง จากผลการคำนวณในรูปที่ 6 นั้น ต้องคำนวณทั้งหมด 6 ครั้ง ซึ่งน้อยกว่าวิธี Bisection คำตอบที่ได้ คือ 0.216281 มากกว่าคำตอบที่ได้จากวิธี Bisection เพียง 0.000033

วิธี Simple Iteration method

การหารากของสมการวิธีนี้ เราจะต้องเปลี่ยนรูปสมการที่จะหา $f(x) = 0$ ให้มาอยู่ในรูป $x = g(x)$ ก่อน เช่น $x^2-4 = 0$ จะเปลี่ยนไปเป็น $x = 4/x$ โดย $g(x) = 4/x$ เป็นต้น จากสมการ $x = g(x)$ เราจะแยกออกเป็นสองสมการคือ

$$y = g(x) \quad \text{----------1}$$
$$y = x \quad \text{----------2}$$

หลังจากนั้นเราจะหาจุดตัดของสมการทั้งสองซึ่งจะเป็นรากของสมการ f(x) หากสมมติค่าเริ่มต้น x0 เป็นรากของสมการ ถ้าแทนค่า x0 ในสมการที่ 1 ก็จะได้

$$y = g(x0) \quad \text{----------3}$$

และเนื่องจากสมการที่ 2 เท่ากับสมการที่ 3 ดังนั้นจึงได้ตามสมการที่ 4

$$x = g(x0) \quad \text{----------4}$$

ค่า x ที่ได้จากสมการที่ 4 นี้จะเป็นรากของสมการค่าใหม่ ซึ่งจะนำมาใช้แทนค่าในสมการที่ 1 ใหม่ สำหรับคำตอบที่แท้จริงจะได้เมื่อค่า x ที่หาได้ใหม่ใกล้เคียงกับค่าเดิมคือแตกต่างกันน้อยกว่าค่า e ที่เรากำหนดไว้

จากรูปที่ 8 เป็นผลการคำนวณรากของสมการ $y = 3x - e^{-2x}$ ซึ่งใช้วิธี Simple iteration ได้คำตอบ x = 0.216280 โดยต้องคำนวณทั้งหมด 15 ครั้ง คำตอบนี้ใกล้เคียงกับที่ได้จากวิธี False positon มาก

วิธี Newton Ralphson method

จากรูปที่ 9 สมมติว่าเราต้องการหารากของสมการ y = f(x) ซึ่งเราลากเส้นตรง l1 ให้สัมผัสกับกราฟของสมการ f(x) ที่จุด x0 และตัดกับแกน x ที่จุด x1

สมมติให้ m คือ ความชันของเส้นตรงดังกล่าวนี้ จะได้ว่า

$$m = f(x0)/(x0 - x1)$$

จากความรู้ทางแคลคูลัสทำให้เราทราบว่า ความชันของเส้นโค้งที่จุด x0 จะเท่ากับค่าอนุพันธ์ลำดับที่ 1 ของ f(x) ที่ x0 คือ

$$m = f'(x0)$$

แต่จากเงื่อนไขที่เราลากเส้นตรง l1 ให้สัมผัสกับเส้นโค้ง f(x) จะได้ว่าความชันทั้งสองค่าเท่ากัน ดังนั้น

$$f'(x0) = f(x0)/(x0 - x1)$$
$$x1 = x0 - (f(x0)/f'(x0))$$

เช่นเดียวกับวิธี Simple iteration ถ้าสมมติให้ x0 เป็นค่าเริ่มต้นในการหารากของสมการเพื่อนำมาหาค่า x1 จากนั้นจะใช้ x1 มาใช้หารากของสมการตัวใหม่ การคำนวณหารากของสมการจะสิ้นสุดเมื่อค่า x ที่หาได้มีค่าใกล้เคียงกับค่าที่หาได้ครั้งก่อน

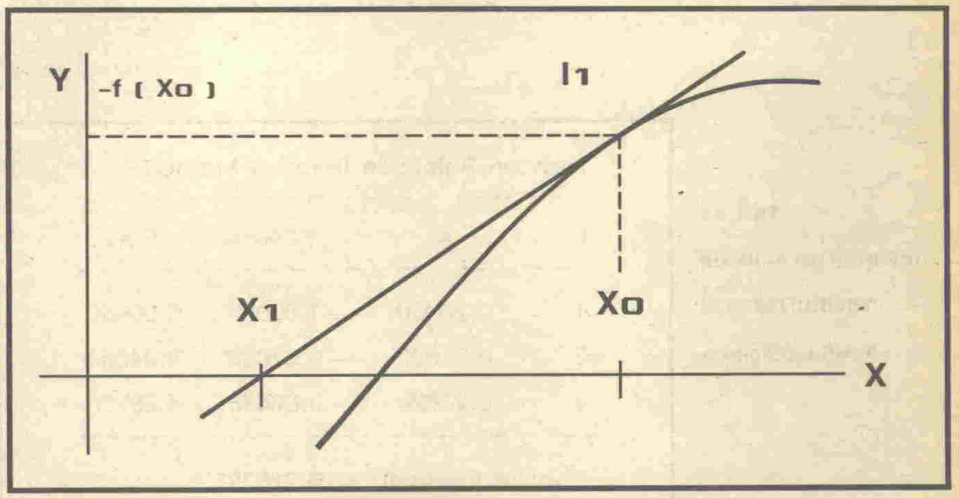
จะเห็นว่าวิธี Newton Ralphson method นี้ เป็นการทดลองลากเส้นตรง

Simple Iteration Method

I	X(I)
1	0.04511
2	0.30458
3	0.18127
4	0.23197
5	0.20960
6	0.21919
7	0.21503
8	0.21682
9	0.21605
10	0.21638
11	0.21624
12	0.21630
13	0.21627
14	0.21628
15	0.21628

Root of Equation = 0.216280

รูปที่ 8 แสดงผลการคำนวณรากของสมการด้วยวิธี Simple iteration



รูปที่ 9 แสดงกราฟของฟังก์ชัน f(x) ซึ่งใช้วิธี Newton-Ralphson

```
#include <stdio.h>
#include <math.h>

float f(float x)
{
    return(3*x - exp(-2*x));
}
float f1(float x)
{
    return(3 + 2*exp(-2*x));
}
float newton(float x1, int n, float e)
{
    float x, fa, f1a;
    int  i;

    puts("Newton-Ralphson Iterative Method");
    puts("---------------------------------");
    puts(" I    X      F(X)    F'(X)");
    puts("---------------------------------");
    /************ Newton-Ralphson Method *************/
    i = 0;
    x = x1;
    do {
     x1 = x;
     fa = f(x1); f1a = f1(x1);
     x = x1 - fa/f1a;
     ++i;
     printf("%3d %8.5f %8.5f %8.5f\n",i,x,fa,f1a);
    } while ((fabs(x-x1) >= e)||(i == n));
    /*****************************************/
    puts("---------------------------------");
    return(x);
}
main()
{
    printf("\nRoot of Equation = %9.6f\n",newton(0,100,0.0001));
}
```

รูปที่ 10 แสดงโปรแกรมซึ่งคำนวณรากของสมการโดยวิธี Newton-Ralphson

ให้สัมผัสกับ f(x) ที่ตำแหน่งรากของสมการนั่นเอง

ข้อเสียของวิธีนี้คือ ต้องหาค่า $f'(x)$ ให้ได้ จึงจะหารากสมการได้ ซึ่งบางครั้งการหา $f'(x)$ นั้นก็ค่อนข้างยาก จากรูปที่ 10 เป็นโปรแกรมที่จะหารากของสมการ $f(x) = 3x - e^{-2x}$ ซึ่ง $f'(x) = 3 + 2e^{-2x}$ และในรูปที่ 11 จะเห็นว่าการคำนวณด้วยวิธีนี้ทำเพียง 3 รอบก็ได้คำตอบ คือ 0.216281 ซึ่งใช้เวลาในการคำนวณน้อย

ก่อนจาก

วิธีการหารากของทั้ง 4 แบบที่ได้นำมาเสนอในบทความนี้เป็นเพียงบางส่วนของวิธีหารากของสมการซึ่งมีอีกหลายวิธี สำหรับวิธีที่ได้เลือกมานี้แต่ละวิธีก็มีความแตกต่างกันออกไป บางวิธีก็ใช้วิธีการคำนวณง่าย ๆ บางวิธีอาจต้องใช้คณิตศาสตร์ชั้นสูงช่วย

นอกจากนี้แล้วในแง่ความเร็วของการคำนวณก็ต่างกันด้วย อย่างไรก็ตาม ถ้าถึงเวลาที่คุณต้องการหารากของสมการ คุณคงสามารถเลือกวิธีใดวิธีหนึ่งจากบทความนี้มาช่วยแก้ปัญหาของคุณได้

รูปที่ 11 แสดงผลการคำนวณรากของสมการด้วยวิธี Newton-Ralphson

```
Newton-Ralphson Iterative Method
---------------------------------
I        X         F(X)      F(X)
---------------------------------
1      0.20000   -1.00000   5.00000
2      0.21620   -0.07032   4.34064
3      0.21628   -0.00035   4.29790
---------------------------------
Root of Equation = 0.216281
```